View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by University of Montana

University of Montana

ScholarWorks at University of Montana

Graduate Student Theses, Dissertations, & Professional Papers

Graduate School

1997

NK fitness functions

Jian Zhang
The University of Montana

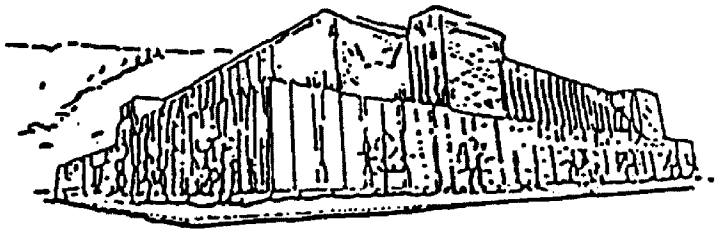
Follow this and additional works at: https://scholarworks.umt.edu/etd

Let us know how access to this document benefits you.

Recommended Citation

Zhang, Jian, "NK fitness functions" (1997). *Graduate Student Theses, Dissertations, & Professional Papers*. 8351.
https://scholarworks.umt.edu/etd/8351

This Thesis is brought to you for free and open access by the Graduate School at ScholarWorks at University of Montana. It has been accepted for inclusion in Graduate Student Theses, Dissertations, & Professional Papers by an authorized administrator of ScholarWorks at University of Montana. For more information, please contact scholarworks@mso.umt.edu.



Maureen and Mike
MANSFIELD LIBRARY

The University of **MONTANA**

Permission is granted by the author to reproduce this material in its entirety, provided that this material is used for scholarly purposes and is properly cited in published works and reports.

** *Please check "Yes" or "No" and provide signature* **

Yes, I grant permission X
No, I do not grant permission ____

Jian Zhang by Alexander C. Wright
Author's Signature

Date 5/21/97

Any copying for commercial purposes or financial gain may be undertaken only with the author's explicit consent.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

NK Fitness Functions

Jian Zhang

B. S., Shandong University, 1984

presented in partial fulfillment of the requirements

for the degree of

Master of Science

in Computer Science

The University of Montana

May 21, 1997

Approved by:

Chairperson

Dean, Graduate School

5-22-97

Date

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

UMI Number: EP39152

All rights reserved

INFORMATION TO ALL USERS
The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript
and there are missing pages, these will be noted. Also, if material had to be removed,
a note will indicate the deletion.

UMI EP39152
Published by ProQuest LLC (2013). Copyright in the Dissertation held by the Author.
Microform Edition © ProQuest LLC.
All rights reserved. This work is protected against
unauthorized copying under Title 17, United States Code

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Jian Zhang, M.S., May 21, 1997 Computer Science

NK Fitness Functions (30 pp.)

Director: Alden H. Wright

In Kauffman's NK model, the fitness function is a sum of terms, each of which depends on K components of the domain string. In the arbitrary model, the positions of the K components in the string are arbitrary. In the adjacent model, the K components are sequential in the string. It has been shown by Richard K. Thompson that there is a polynomial-time algorithm for the adjacent model, In this work, an approximation algorithm for the NK fitness optimization problem as well as the approximation threshold of the algorithm are given. It is also shown that this problem is MAXSNP-complete, but there is no polynomial-time approximation scheme for it. The NK fitness function under bandwidth constraint is discussed and it is shown that for any function f such that $f(n) \geq \log n$, $\mathrm{NK}_{Fitness}(f(n))$ is log-space complete for $N_{TISP}(POLY, f(n))$, the class of problems solvable by polynomial-time nondeterministic algorithms that use simultaneously at most $f(n)$ space when restricted to bandwidth $f(n)$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table of Contents

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 1

Introduction

1. Introduction to the NK Fitness Functions

The concept of a "rugged fitness landscape" was introduced by Sewall Wright. It is a term for a mapping of the vertices of a finite graph to the real numbers; it represents the abstract "fitness" of various kinds of organisms in various contexts. This analogy has been used in many evolutionary biological and physical models in several fields ever since.

Later, Stewart Kauffman (see [10]) introduced the NK Model of Random Epistatic Interactions. This is a simple stochastic model of a fitness landscape and it can be used as an aid to the qualitiative understanding of more complex models. This model defines a specific stochastic algorithm for describing a fitness function on the domain of strings. (From the point of view of optimization, a fitness function is simply an objective function that is to be maximized.) The parameter K determines the "ruggedness" of the fitness landscape. For $K = 1$, the function is linear (non-epistatic), and for $K = N$, the function is random, or "uncorrelated".

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2. Description of the NK Model

In nature, one interpretation of the fitness function of an organism is that it is a function of genome of the organism, in other words a function of the set of all possible sequences of the four nucleotide bases. Nature evaluates this function by translating these nucleic sequences into sequences of amino acids. An interpretation of the NK model ignores genetics and assigns a fitness to the sequences of amino acids directly. The NK model makes two further simplifications: the first, that the amino acid sequence has a fixed length of N sites; and the second, that there are only two possible amino acids, rather than the full complement of twenty, that can occupy a given site of the sequence. This last assumption is justified by the fact that almost all of the properties of such sequences are determined by their three dimensional, folded structures, which in turn are determined by the chemical properties of the constituent amino acids.

The most important of those properties is polarity: those amino acids in the sequence that are polar get pulled to the outside of the folded structure by chemical attraction to surrounding water molecules, and non-polar amino acids get pushed into the interior of the structure

The assumption of two amino acids per site also dramatically simplifies the modelling task by reducing the argument of the fitness function to a bit string, let us say, **a**. The NK model assigns a real valued "fitness" to **a** by first assigning a real valued "fitness contribution", f_i, to the ith bit, a_i, in **a**. Each such assignment depends, not just on i and the value of a_i, but also on K other bits where $1 \leq K \leq N$, which we call "neighbors". The fitness contribution of each site is a random function, $f_i(s_i)$, of the substring s_i, which is formed by the ith bit and its k neighbors.

Now, we will formally introduce the abstract model of the "fitness landscape"

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

problem in a more precise way.

3. Definition of the NK Fitness Functions

To discuss various applications of NK Fitness Funtion is beyond the scope of this paper. We will focus on the study and analysis of the abstract model for the fitness landscape problem. We will use the definition in [7] to represent the NK model in a mathematically precise fasion. To discuss various applications of NK Fitness Funtion is beyond the scope of this thesis.

Let Σ be a fixed alphabet. An NK fitness function f is a funtion over string $\mathbf{a} \in \sum^N$. It is the sum of N terms, where each term depends on K components of the string. (Under Kauffman's definition of K, each term i depends on $K+1$ components of the string, but we will use our definition since it is more natural). In other words,

$$f(\mathbf{a}) = \sum_{i=0}^{N-1} f_i(\mathbf{a})$$

where each f_i depends on K components of $\mathbf{a}$, including component a_i. Thus, f_i can be defined by a table of size 2^K. The NK model further specifies that the values in this table are chosen randomly from a uniform distribution over the interval $[0, 1]$.

Component a_i will always be used in deciding the value of each f_i. Depending on the way we choose the rest of the components, we can define two different models: We can choose the bits that are adjacent to component a_i; then it will form the *adjacent* model. On the other hand, we can arbitrarily choose the remaining $K-1$ components, in which case we will have the *arbitrary* model.

A fitness function over string $\mathbf{a} \in \sum^N$ is *linear* or *non-epistatic* if it is a linear (or affine) function of the components of the string $\mathbf{a}$. In other words, the fitness function $f : \sum^N \longrightarrow R^+$ is linear if

$$f(\mathbf{a}) = c + \sum b_i a_i$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

where $c > 0$ and b_i are real numbers, and a_i denotes the ith component of string **a**. The NK model with $K = 1$ gives a linear fitness function.

To illustrate the NK model, we present a simple example of an NK fitness function f (*the adjacent* model) over the alphabet $\{0, 1\}$ where $N = 4$ and $K = 2$. (This example comes from [9].)

Let

$$f(\mathbf{a}) = f_0(a_0, a_1) + f_1(a_1, a_2) + f_1(a_2, a_3) + f_1(a_3, a_0)$$

where f_i is defined as:

a_i	a_{i+1}	f_0	f_1	f_2	f_3
0	0	1	3	1	2
0	1	4	2	1	3
1	0	1	2	4	2
1	1	3	2	4	5

If the string **a** is 0110, then

$$f(0110) = f_0(0, 1) + f_1(1, 1) + f_2(1, 0) + f_3(0, 0) = 4 + 2 + 4 + 2 = 12$$

.

4. More about the Definition

The *arbitrary* model seems more complicated than the *adjacent* model and we will have more detailed discussion about these models later in the chapter. They are mathematically defined later in the chapter.

Also, sometimes we would have to consider the corresponding NK fitness decision problem:

Definition 1.1 *The NK fitness decision problem: Given an NK fitness function F*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

and a positive value V, does there exist a string x such that

$$F(x) \geq V?$$

Finally, since we will also discuss the corresponding optimization problem, we will give the definition here:

Definition 1.2 *The NK fitness optimization problem (we will call it NKOPT in this thesis): Given an NK fitness function F, find x so that*

$$F(x) \geq F(y)$$

for all strings y.

Obviously, once the NK fitness optimization problem is solved, the output for the corresponding NK fitness decision problem is determined, too. Since the optimum value of the NK fitness function for the *adjacent* model can be determined in polynomial time using the algorithm given in [9], the decision problem can also be solved in polynomial time for the *adjacent* model. There are more discussions about the complexity of the adjacent model problem later in this thesis.

In this work we consider an extension of Kauffman's NK model. As mentioned above, we change the interpretation of the parameter K. Under our interpretation, each component of an NK fitness function is affected by K components of the string rather than $K + 1$ components. We also remove the assumption that the values in the table that defines a component of the fitness function are chosen randomly, and instead assume that these values are arbitrary non-negative integers. We use integers instead of real numbers since we are interested in computational complexity, and an integer representation is easier to deal with. Under Kauffman's model, the sizes of the N additive terms of the fitness function are statistically identical; we make no such assumption.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

To specify an NK fitness function $f = \sum_{i=0}^{N-1} f_i$, we must specify the positions that influence each term f_i, and we must specify the f_i functions (see [7,9]).

We specify the positions through projection functions $p_0, p_1, \ldots, p_{N-1}$, where each p_i is a mapping from $\sum^N$ to $\sum^K$. Each p_i is defined by a K-subset of $\{0, 1, \ldots, N-1\}$. For example, if $K = 3$ and p_2 is defined by the subset $\{1, 3, 6\}$, then

$$p_2(a_0, a_1, \ldots, a_{N-1}) = (a_1, a_3, a_6)$$

The difference between the *adjacent* model and *arbitrary* model is the way we choose the positions. The *adjacent* NK model is the simpler to specify, the subset of $0, 1, \ldots, N-1$ that defines p_i is $\{i, i+1, \ldots, i+K-1\}$, where the indices are taken modulo N. For the *arbitrary* NK model, the subset that defines p_i is arbitrarily chosen (except that i is always in this subset). We assume that the component functions f_i map $\sum^K$ into N. Then f is written more precisely as

$$f = \sum_{i=0}^{N-1} f_i \circ p_i$$

where the domain of each f_i is $\sum^K$.

We assume that each of the functions f_i is defined by a table which associates each K-tuple from $\sum^K$ with a integer (see [7,9]).

In [7, 9], it has been shown that the NK fitness function decision problem for the *adjacent* model belongs to the class P, which means that there is a polynomial-time algorithm for the *arbitrary* model NKOPT problem. It also has been shown that the analogous NK problem for the *arbitrary* model is NP-complete.

Since it s unlikely that $P = NP$, this means it is also unlikely that we can find a polynomial-time algorithm for the NKOPT for the *arbitrary* model. In Chapter 2, we will try to find the next best thing: A polynomial-time approximation algorithm.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 2

A Polynomial-time Approximation Algorithm

1. Definitions

It has been shown in [7, 9] that the NKOPT for the *arbitary* model is NP-complete. Furthermore, it not only belongs to the class of NP-complete problems, as will be shown later, it also belongs to another interesting family of complexity problems.

Although all NP-complete problems share exponential worst-case complexity (unless $P = NP$), they have little else in common. When seen from almost any other perspective, they have interesting, unique diversity. In this paper, we want to asseses the difficulty of NKOPT. We will accomplish this by studying the problem against several complexity classes [3].

An NP-completeness proof is typically the first act in the analysis of a computational problem by the methods of the theory of algorithms and complexity, not the last. Once NP-completeness has been established, we are not expecting to solve the problem exactly, efficiently, every time, since it is not likely that we will be

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

able to solve the problem in polynomial time. If we are dealing with an optimization problem, we would rather study the possibility of a "quick-and-dirty" algorithm which return feasible solutions that are not necessarily optimal, but the time consumed in the process is rather short (say, bounded by a polynomial in the input size). Such heuristics can be empirically valuable methods for attacking an NP-complete optimization problem even when nothing can be proved about their worst-case (or expected) performance. In some fortunate cases, however, the solutions returned by a polynomial-time heuristic are guaranteed to be "not too far away from the optimal."

Definition 2.1 *Suppose that A is an optimization problem. This means that for each instance x we have a set $F(x)$ of* feasible solutions, *and for each solution $s \in F(x)$ we have a positive integer cost $c(s)$ (we use the term* cost *and notation $c(s)$ even in the case of maximization problems). The optimal cost is $OPT(x) = min_{s \in F(x)} c(s)$ (or $max_{s \in F(x)} c(s)$, if A is a maximization problem). Let M be an algorithm which, given any instance x, returns a feasible solution $M(x) \in F(x)$. We say that M is an ϵ-approximation algorithm, where $\epsilon > 0$, if for all x we have (see [3])*

$$\frac{|c(M(x)) - OPT(x)|}{max\{OPT(x), c(M(x))\}} \leq \epsilon$$

Intuitively, a heuristic is ϵ-approximate if the "relative error" of the solution found is at most ϵ. For a maximization problem, an ϵ-approximate algorithm returns solutions that are never smaller than $1 - \epsilon$ times the optimum. For minimization problem, the solutions returned are never more than $(1 - \epsilon)^{-1}$ times the optimum. Evidently, the ϵ here is used to measure how far away the approximate solution is from the optimum.

For each NP-complete optimization problem A we shall be interested in determining the smallest ϵ for which there is a *polynomial-time* ϵ-approximation algorithm for A. Sometimes no such ϵ exists, but there are also approximation algorithms

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

that achieve arbitrarily small error ratios, which is not too far away from an optimal solution.

Definition 2.2 *The* approximation threshold *of A is the greatest lower bound of all $\epsilon > 0$ such that there is a polynomial-time ϵ-approximation algorithm for A.*

The approximation threshold of an optimization problem can be anywhere between zero (arbitrarily close approximation is possible) and one (essentially no approximation is possible) (see [3]). Of course, any optimization problem that has a polynomial-time algorithm has approximation threshold zero.

As mentioned above, it has been shown in [7, 9] that the NKOPT for the arbitarry model is NP-complete, which means that it is expected to be impossible to find a polynomial-time algorithm for it. This is bad news, but it does not mean that this is the end of the world. In fact, for many problems, this is just a new start.

In the following section, we will show some positive results by finding the next best thing: a polynomial-time approximation algorithm for the NKOPT.

2. The Approximation Algorithm

This problem is similiar to the MAXSAT problem and we can solve it in a similiar way:

There are N bits in the input string. Since each bit could only be 0 or 1, there are 2^N possible bit assignments. If we calculate the value of the NK fitness function corresponding to each assignment, what is the average value of these values?

We will denote the set of all N-bit strings as S, the average value of function f on the set S as $AVG(f^S)$, and the average value of $f_i \circ p_i$ on the set S as $AVG(f_i^S)$.

From the definition of NK fitness function, we have

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$f = \sum_{i=1}^{N-1} f_i.$$

Therefore, we have

$$AVG(f^S) = \sum_{i=0}^{N-1} AVG(f_i^S).$$

To calculate $AVG(f^S)$, we have to remember that each function f_i depends on exactly K bits of the string; therefore, if we identify the K-bit strings with the integers from 0 to $2^K - 1$, there are only 2^K different values:

$$f_i[j] \text{ for } j = 0, 1, \ldots, 2^K - 1$$

for each function. Then we have

$$AVG(f_i^S) = \frac{1}{2^K} \times \sum_{j=0}^{2^K-1} f_i[j].$$

Let m_i be the maximum value of f_i. Since the sum of a sequence of nonnegative numbers is always greater than its maximum, we have

$$\sum_{j=0}^{2^K-1} f_i[j] \geq m_i$$

We may thus obtain a lower bound for $AVG(f_i^S)$:

$$AVG(f_i^S) \geq \frac{1}{2^K} \times m_i$$

Let the optimal value of f be MAX, then it is obvious that $\sum_{i=0}^{N-1} m_i \geq MAX$, and we have the following conclusion:

Lemma 2.3 *The average value of the NK fitness function satisfies*

$$AVG(f^S) \geq \frac{1}{2^K} \times MAX.$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Using the result above, we will try to reduce the size of the string set to 1 without reducing the average value of the NK fitness function corresponding to the set; this will allow us to find an approximate solution for the NK fitness optimization problem.

Suppose that we set bit 0 to 1 in all string assignments; then we have a set $S1$ of string assignments which only involve bit 1 through bit $N-1$, and we can again calculate the average value $AVG(f^{S1})$ of the NK fitness function for this set of string assignments. Similiarly if we set bit 0 to 0, then we have a set $S0$ of string assignments. Let the average of f for this set be $AVG(f^{S0})$. Since sets $S0$ and $S1$ have the same size, now it is very easy to see that

$$AVG(f^S) = \frac{1}{2}(AVG(f^{S1}) + AVG(f^{S0}))$$

This equation shows that, if we set bit 0 to 0 when $AVG(f^{S1}) < AVG(f^{S0})$ and 1 otherwise and we replace S by $S0$ in the first case ($S1$ in the second), then we end up with a string set with average value at least as large as the original $AVG(f^S)$.

We can continue like this, always splitting the string into two subsets and assigning to the next bit the value that maximizes the average function value on the resulting string set. In the end, all bits have been assigned values. However, since our average value never decreases in the process and the last set we have will only have one member left, we know that the value of the NK fitness function corresponding to the final string (since now we have only one string in the set, it is the same as the average value of the set now) is at least as large as

$$\frac{1}{2^K} \times MAX.$$

These remarks suggest the following algorithm for approximating a solution for the NK fitness optimization problem.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

NK_OPTIM(S)

/// s is the approximately optimal string.

/// S is initialized to the set of all N-bit strings

/// and evolves as the bits of S are assigned;

/// eventually, $S = s$.

for i **from** 0 **to** $N-1$ **do**

$S0 \leftarrow$ subset of S where the ith bit is 0

$M0 \leftarrow$ average of f over $S0$

$S1 \leftarrow$ subset of S where the ith bit is 1

$M1 \leftarrow$ average of f over $S1$

if $M0 > M1$ **then**

$s[i] \leftarrow 0$

$S \leftarrow S0$

else

$s[i] \leftarrow 1$

$S \leftarrow S1$

end /// The approximately optimal string is now in s

return s

Algorithm 1

Notice that inside the "for" loop, each step needs to calculate at most $2^K \times N$ values (each f_i depends on at most K bits). Therefore, the complexity of the algorithm is

$$N \times 2^K \times N = N^2 \times 2^K,$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

which is a polynomial in N.

Now we have an approximation threshold for the algorithm:

Theorem 2.4 *The approximation threshold for the algorithm with $K \geq 2$ is at most $1 - 1/2^K$.*

Proof: The approximation threshold for the algorithm is at most

$$\frac{MAX - \frac{1}{2^K} \times MAX}{MAX} = 1 - 1/2^K.$$

□

In fact, the string given by this algorithm is no more than average, but nothing more than that is guaranteed. There might be a way, such as hillclimbing, to optimize it, but no better approximation threshold has been found so far.

The MAXGSAT problem is a well-known optimization problem which can be reduced to the NKOPT problem. In the k-MAXGSAT problem, we are given a set of Boolean expressions, each involving at most k Boolean variables. The problem is to find an assignment of values to the Boolean variables that maximizes the number of satisfied expressions. An instance of the k-MAXGSAT problem can be reduced to an instance of the NKOPT problem with the $K = k + 1$ (where K is the paramter of NKOPT instance). The best known approximation threshold for the k-MAXGSAT problem is $1 - 2^{-k}$ (see [3]), so it would seem to be a difficult problem to find a substantially better approximation threshold for the NKOPT propblem.

3. Classify the Problem by Approximation Threshold

Depending on how large the approximation threshold is, we can roughly divide the approximation problem into three categories (from the most difficult to the least difficult):

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1. The approximation threshold for the problem is 1. Many approximation problem fall into this category. The unrestricted INDEPENDENT SET and CLIQUE problems belong to this category, but they are right on the edge because we have the following result from [3]: unless $P = NP$, the approximation threshold of INDEPENDENT SET and CLIQUE is one. As we mentioned before, it is thought to be impossible to do any polynomial-time approximation for these problems.

2. The approximation threshold for the problem is at most $\varepsilon \in (0, 1)$. Some well-known problems such as NODE COVER, MAXSAT and Maximum Cut belong to this category. This is another common case among the approximation problems. However, there is one major difference between this case and the last. Actually, as will be mentioned in Chapter 4, there is a special class called MAXSNP, and any approximation problem with a threshold less than one belongs to the class MAXSNP. We will have more discussion about this class in Chapter 4.

3. The approximation threshold for the problem is 0. That is, for any $\varepsilon > 0$ there is a polynomial ε-approximation algorithm for the problem. There is a sequence of algorithms whose error ratios have limit 0. In other words, this is the kind of problem for which approximation can be arbitrarily close. The KNAPSACK problem is one example. This category is evidently the "easiest" among these three categories. It is the best thing we can find that is close to a polynomial-time algorithm for the problem.

Since we have already shown that the approximation threshold for the NK fitness problem is at most $1 - \frac{1}{2^K}$, this problem might belong to the second category or the third, depending on whether the approximation is limited or unlimited. This will be

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the topic for the next two chapters.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 3

MAXSNP-Completeness

1. Definition

When we ask whether a problem has a polynomial-time algorithm to solve it, the class NP is introduced. When we ask whether a problem has a polynomial-time approximation scheme, a new class MAXSNP is introduced (see [3]). In both cases, for several natural problems we asked an important and difficult-to-answer question (earlier, whether the problems have a polynomial-time algorithms, now whether they have polynomial-time approximation schemes). There are a number of similarities between the developments of these two classes.

For the class NP, a reduction is introduced to define the relationship among the problems in the class. We also have a new reduction that is parallel to the reduction for NP problems (see [3])

Definition 3.1 *An* L-reduction *from A to B is a pair of functions R and S, both LOGSPACE-computable, with the following two additional properties:*

1. *If x is an instance of A with optimal cost $OPT(x)$, then $R(x)$ is an instance of B with optimal cost that satisfies*

$$OPT(R(x)) \leq \alpha \cdot OPT(x),$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

where α is a positive constant.

2. *If s is any feasible solution of $R(x)$, then $S(s)$ is a feasible solution of x such that*

$$|OPT(x) - c(S(s))| \leq \beta \cdot |OPT(R(x)) - c(s)|,$$

where β is another positive constant particular to the reduction (and we use c to denote the cost in both instances).

That is, S is guaranteed to return a feasible solution of x which is not much more suboptimal than the given solution of $R(x)$. Notice that, by the second property, an L-reduction is a true reduction; if s is the optimal solution of $R(x)$, then indeed $S(s)$ must the optimal solution of x (see [3]).

Definition 3.2 *A in class $MAXSNP_0$ is defined in terms of the expression*

$$max_S|(x_1, \ldots, x_k) \in V^k : \phi(G_1, \ldots, G_m, S, x_1, \ldots, x_k)|$$

The input to a problem A is a set of relations

$$G_1, \ldots, G_m$$

over a finite universe V. We are seeking a relation $S \subset V^r$ such that the number of k-tuples $(x_1, \ldots, x_k)$ for which ϕ holds is as large as possible.

Definition 3.3 *MAXSNP is the class of all optimization problems that are L-reducible to a problem in $MAXSNP_0$.*

Definition 3.4 *An optimization problem is MAXSNP-complete if it is in MAXSNP and all the problems in MAXSNP can be L-reduced to this problem.*

In fact, there are many well-known NP-complete problems that belong to the MAXSNP-complete class, here are some examples (see [3]):

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1. MAX3SAT;
2. 4-DEGREE NODE COVER;
3. 4-DEGREE INDEPENDENT SET;
4. 5-OCCURENCE MAX2SAT;
5. MAX NAESAT;
6. MAX CUT.

Since we will need to use MAX3SAT as an example thoughout the paper, the definition of this problem is given here:

Definition 3.5 *Given a Boolean formula in conjunctive normal form (CNF) with at most* 3 *literals per clause, maximize the number of true clauses (A* 3*-CNF formula is a conjunction of clauses of the form* $u_i \vee u_j \vee u_k$*, where* u_i, u_j, u_k *are literals. A literal is either a Boolean variable or the negation of a Boolean variable.*

An example of a 3-CNF formula is:

$$(x_2 \vee \overline{x_3} \vee x_4) \wedge (\overline{x_2} \vee x_4)$$

Now, if we want to show an optimization problem is MAXSNP-complete, it suffices to show that it is in MAXSNP and the MAX3SAT problem can be L-reduced to this problem. Since we already know that all the problems in MAXSNP can be L-reduced to MAX3SAT, this means that they can also be L-reduced to this problem. Thus, both conditions in the definition of MAXSNP-complete class will be satisfied.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2. Is the NK Fitness Problem MAXSNP-complete?

Since we can see many similarities between the NK fitness problem and the SAT problem (family, maybe we should say), and MAX3SAT is MAXSNP-complete, we have some reasons to believe that NK fitness problem is also MAXSNP-complete.

Theorem 3.6 *The arbitarily model NKOPT is MAXSNP-complete.*

Proof:

A result given in [3] shows that if a problem has a polynomial-time approximation algorithm with threshold less than 1, then the problem is in MAXSNP. Using the result from Theorem 2.2 of Chapter 2, it follows that the NK fitness problem is in MAXSNP.

To show this problem is MAXSNP-complete, we will show that MAX3SAT can be L-reduced to this problem.

Given a 3-CNF formula, we will construct a corresponding NK fitness function f over the alphabet $\{0, 1\}$ so that the value of f is the number of true clauses in the CNF formula. Let $M1$ be the maximum number of variables in the 3-CNF formula and let $M2$ be the number of clauses in the 3-CNF formula; then let $N = \max\{M1, M2\}$.

Then we will reduce MAX3SAT to the NK-fitness function problem for $K = 4$. We defined f_i as follows

If the ith variable is not in the ith clause, then we will add the ith variable into the f_i definition table; after that, we will add arbitrary variables to the f_i definition so that there will be 4 variables in the table.

If we have more variables than clauses, then for $i > M2$, we define $f_i = 0$ identically.

We assume that the ith Boolean variable of the formula is associated with the ith string position, and that the ith clause of the formula is associated with f_i. Thus,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

each f_i depends on three string positions. Let a string position value of 1 correspond to a *true* value of the corresponding Boolean variable, and a value of 0 correspond to *false*. The value of f_i will be defined to be 1 if the corresponding clause of the formula is *true* and 0 if the corresponding clause is *false*.

In the above example 1., f_1 corresponds to the clause $(x_2 \vee x_3 \vee \overline{x_4})$; thus $(x_2 \vee x_3 \vee \overline{x_4})$ will depend on string positions $2, 3, 4$. As suggested above, we will add position 1 to it, which leads to the following table defining f_i

a_1	a_2	a_3	a_4	f_1
0	0	0	0	1
1	0	0	0	1
0	0	0	1	1
1	0	0	1	1
0	0	1	0	0
1	0	1	0	0
0	0	1	1	1
1	0	1	1	1
0	1	0	0	1
1	1	0	0	1
0	1	0	1	1
1	1	0	1	1
0	1	1	0	1
1	1	1	0	1
0	1	1	1	1
1	1	1	1	1

For f_2, a_2 is already there, so we arbitrarily add 2 variables, say a_1, a_3 to it and define it analogously.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Since we do not have four clauses, we add two functions f_3, f_4, and define them to be identically zero.

Clearly, the value of $f = \sum_{i=1}^{4} f_i$ is the number of *true* clauses in the 3-CNF formula.

It is also clear that the reduction is an L-reduction since the optimal values in the two instances are the same. □

It seems like the question of whether MAXSNP-complete problems have polynomial-time approximation schemes is very much like the $P = NP$ question. As it happens, the similarity runs a little deeper than anyone had expected. A sequence of deep and striking recent results has established that the two questions are equivalent: MAXSNP-complete problems have polynomial-time approximation schemes if and only if $P = NP$ (see [3]); naturally, this is the strongest negative result for approximation that we could have expected. We will have more detailed discussion in later chapters.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 4

Polynomial-time Approximation Scheme

Definition 4.1 A polynomial-time approximation scheme *for an optimization problem A is an algorithm which, for each $\epsilon > 0$ and instance x of A, returns a solution with a relative error of at most ϵ, in time bounded by a polynomial (depending on ϵ) in $|x|$, where $|x|$ denotes the length of an encoding of x.*

If the running time depends polynomially on $\frac{1}{\epsilon}$ as well, the approximation scheme is called fully polynomial. This is the case, e.g., with the KNAPSACK problem, whose running time is bounded by $O(\frac{n^3}{\epsilon})$.

If we could find a polynomial-time approximation scheme for the NK fitness optimization problem, since it is also MAXSNP-complete, this would imply that all the MAXSNP problems have polynomial-time approximation schemes. Since nobody had been able to establish the latter, it is unlikely that we can find a polynomial-time approximation scheme for NKOPT. We will prove there is no such scheme unless $P = NP$.

In [3], there are some discussions about MAXSNP and MAXSNP-completeness. The most interesting results are given below; these further illustrate the similarity

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

between NP-complete problems and MAXSNP-complete problems.

(1) NKOPT is MAXSNP-complete (from Chapter 3).

(2) MAXSAT is in MAXSNP.

(3) If a MAXSNP-complete problem has a polynomial-time approximation scheme, then every problem in MAXSNP has one.

(4) If there is a polynomial-time approximation scheme for MAXSAT, then $P = NP$.

Following from these results, we have:

Theorem 4.2 *Unless $P = NP$, there is no polynomial-time approximation scheme for the NK Fitness problem.*

Proof: If there is a polynomial-time approximation scheme for the NK Fitness problem, then from (1) and (3), all problems in MAXSNP have polynomial-time approximation schemes. Since the MAXSAT problem is in MAXSNP, this means that there is a polynomial-time approximation scheme for MAXSAT; then from (4), we have $P = NP$. □

In fact, we have a stronger result, which may be proved in the same fasion.

Theorem 4.3 *Unless $P = NP$, there is no polynomial-time approximation scheme for any MAXSNP-complete problem.*

From these results, we can clearly see the parallelism between the NP class and MAXSNP class. If any NP-complete problem has a polynomial-time algorithm to solve it, then any MAXSNP-complete problem has an approximation scheme. On the other hand, if any MAXSNP-complete problem has an approximation scheme, then any problem in class NP has a polynomial-time algorithm, which means $P = NP$. In other words, to prove that a MAXSNP-complete optimization problem has a polynomial-time approximation scheme is equivalent to proving that $P = NP$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 5

NK Fitness Function with Bandwidth Constraints

In this chapter, we will switch our emphasis from the NK fitness optimization problem to the corresponding decision problem. With the help of bandwidth restriction, we will establish some more facts about the NK fitness problem.

Bandwidth restrictions are considered in [2] for several NP-complete problems, including 3SAT, VERTEX COVER and HAMILTONIAN CIRCUIT. It is shown that these problems when restricted to graphs, formulae, sets of triples, etc., of bandwidth $f(n)$ are LOGSPACE-hard for the class of problems solvable by polynomial-time nondeterministic algorithms that use simultaneously at most $f(n)$ space. This class is denoted by $N_{TISP}(POLY, f(n))$.

Definition 5.1 *Let $G = (V, E)$ be a finite undirected (or directed) graph. A one-to-one function L mapping the set of vertices V onto the set $\{1, 2, \ldots, |V|\}$ is a numbering or a 'linear layout' of G. The graph G has bandwidth k under L, denoted $b(G, L) = k$, if*

$$k = max\{|L(x) - L(y)| : x, y \text{ is an edge of } G\}.$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

G has bandwidth k, denoted $b(G) = k$, *if*

$$k = min\{b(G, L) : L \text{ is a linear layout of } G\}.$$

We extend the concept of bandwidth to several other structures. For example, a well-formed formula $w = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ in 3-conjunctive form has bandwidth k if k is the least integer such that, whenever a variable x or its negation appears in clauses C_i and C_j, the inequality $|i - j| \leq k$ is satisfied. This means, for any well-formed formula w, let $G(w)$ be the graph which has one vertex for each clause in w and an edge connecting vertices when they correspond to clauses containing the same variable or its negation. Then $G(w)$ has bandwidth k (under the linear layout induced by the order of clauses in w) if and only if w has bandwidth k (under the numbering given by the order of the clauses).

There is another natural definition for bandwidth in well-formed formulae. Instead of having a linear layout of the clauses, one can consider a linear layout of the variables. Given such a layout L of the variables, w has this 'type-2 bandwidth' k if, for all variables x and y appearing in the same clauses, $|L(x) - L(y)| \leq k$.

For the $\text{NK}_{Fitness}$ problem, we will use the following bandwidth definition:

Definition 5.2 *An* $NK_{Fitness}$ *problem has a bandwidth k if whenever bit i and bit j appear in clauses any term of the* $NK_{Fitness}$ *function, the inequality* $|i - j| \leq k$ *is satisfied.*

For convenience, we shall denote a problem A restricted to structures which have bandwidth $f(n)$ under the order of input (layout) by $A(f(n))$. For example, the problem $\text{NK}_{Fitness}$ restricted to positions which have bandwidth $f(n)$ under the layout induced by the order of the input will be denoted by $\text{NK}_{Fitness}(f(n))$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Definition 5.3 *Let T be a LOGSPACE-computable function which maps instances of a problem A into instances of a problem B. We say that T is* bandwidth preserving *if there exists a constant $c > 0$ such that, for any instance x of problem A, $T(x)$ is an instance of problem B with bandwidth at most c times the bandwidth of x.*

Bandwidth preservation is an attribute to be applied to LOGSPACE reductions. We say that a problem A is *LOGSPACE reducible to a problem B by a bandwidth preserving reduction*, denoted $A \leq_{log}^{bw} B$, if there is a LOGSPACE-computable function T that is bandwidth preserving such that, for all instances x of problem A, x is a positive instance of problem A if and only if $T(x)$ is a positive instance of problem B. It is easy to see that the relation $\leq_{log}^{bw}$ is transitive.

In [2], it was shown that $3SAT(f(n))$ restricted to well-formed formula with bandwidth $f(n)$ is LOGSPACE complete for the class of problems that can be solved in polynomial-time by nondeterministic algorithms that use at most $f(n)$ worktape space, where n is the length of the input. This simultaneous nondeterministic polynomial time and $f(n)$ space class is denoted by $N_{TISP}(POLY, f(n))$.

Now, we will switch our attention to the NK fitness decision problem, denoted by $\text{NK}_{Fitness}$. To show this problem is LOGSPACE complete for $N_{TISP}(POLY, f(n))$, we need to show two things:

1. $3SAT \leq_{log}^{bw} \text{NK}_{Fitness}$.

2. $\text{NK}_{Fitness}(f(n))$ is in $N_{TISP}(POLY, f(n))$.

Fortunately, these are both true.

Theorem 5.4 $3SAT \leq_{log}^{bw} NK_{Fitness}$.

Proof: The reduction is the same as that described in Chapter 3; we only add a layout for bandwidth considerations.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

It is obvious that the bandwidth of the constructed NK fitness function remains the same as in the $3SAT$ problem; thus $3SAT \leq_{log}^{bw} \mathrm{NK}_{Fitness}$. □

Theorem 5.5 *$NK_{Fitness}(f(n))$ is in $N_{TISP}(POLY, f(n))$.*

Proof: The exsitence of a nondeterministic polynomial-time algorithm is equivelent to the existence of a polynomial-time verification algorithm.

A verification algorithm (see [8]) is a two-argument algorithm A where one argument is an ordinary input string x and the other is a binary string y called a certificate. A verifies an input string x if there exists a certificate y such that $A(x, y) = 1$.

If the problem and the certificate are encoded over an alphabet Σ, then the set of encodings of problems verified by A is

$$\{x \in \Sigma^* \ : \ \exists y \in \Sigma^* \text{ such that } A(x, y) = 1\}$$

In the case of the NK fitness function decision problem with parameter V, the certificate will be a string y such that $F(y) \geq V$.

Given the string x which encodes the function F and the parameter V, and given the string y, the algorithm will need to verify that $F(y) \geq V$ whenever x encodes a problem whose answer should be "yes". Since the input string x must include a specification of the function F by means of its definition tables, where there are N tables, each of size 2^K. The algorithm can compute $F(y)$ by adding the results of N table lookups. Thus, the algorithm requires only constant space. □

The next theorem follows from the previous two theorems.

Theorem 5.6 *For any function f such that $f(n) \geq \log n$, $NK_{Fitness}(f(n))$ is log space complete for $N_{TISP}(POLY, f(n))$.*

Many well-known problems are discussed in [2] and it has been shown that the following problems are LOGSPACE complete for $N_{TISP}(POLY, f(n))$ under a bandwidth $f(n)$ constraint:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1. 3SAT;
2. VERTEX COVER;
3. INDEPENDENT SET;
4. HITTING SET;
5. 3-DIMENSIONAL MATCHING;
6. EXACT COVER BY 3 SETS;
7. PARTITION INTO TRIANGLES;
8. PARTITION INTO PATHS OF LENGTH 2;
9. GRAPH 3-COLORABILITY;
10. SIMPLE MAX CUT;
11. GRUNDY NUMBERING.

Many well-known NP-complete problems are in $N_{TISP}(POLY, f(n))$, but it doesn't seem likely that $N_{TISP}(POLY, f(n))$ is contained in P for any function f that grows more rapidly than a logarithm. Consequently, all the above problems, including the NK fitness problem, are thought to be intractable for functions that grow more rapidly than a logarithm. Of course, these problems are in P when the bandwidth is restricted to $\log n$, since $N_{SPACE}(\log n)$ is a subset of P (see [2]). The NK fitness problem (*arbitrary* model) is in $N_{TISP}(POLY, f(n))$ when restricted to positions with bandwidth $f(n) \leq \log n$ because, like the *adjacent* model, the effect of each position is restricted to only a certain range.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 6

Conclusions and Further Study

1. Conclusions

In this thesis, we studied the NK fitness function from a complexity point of view. From [7, 9], we learned that the NK fitness optimization problem for the *adjacent* model is in P, while the NK fitness decision problem for the *arbitrary* model is NP-complete.

We paid special attention to the *arbitrary* model. As we mentioned before, the NP-completeness of NKOPT is yet a new start for the study of NK fitness functions. We found an polynomial-time approximation algorithm which has an approximation threshold of $1 - 2^{-K}$. In [3], it was claimed that the best known approximation algorithm for the k-MAXGSAT problem has an approximation threshold of $1 - 2^{-k}$. Since the k-MAXGSAT problem reduces to the k+1-NKOPT problem, it is reasonable to expect that $1 - 2^{-K}$ might be the best approximation threshold for K-NKOPT, though this certainly still awaits proof.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2. Further Study

The fitness landscape model has been widely studied in various fields. The complexity of the model is important for the applications. The approximation of the model is even more important in practice. The approximation algorithm we presented is only one possibility and there might be a better algorithm awaiting discovery.

Another direction we could turn would be to study the NK fitness problem for s specific values of K. With this kind of restriction, it is likely that one could obtain better results. Since in real applications, the values of K for specific models are usually small, this approach might really be worth the effort.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Bibliography

[1] Bart Selman, Hector Levesque and David Mitchell. *A New Method for solving Hard Satisfiability Problems*, Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92) San Jose, CA, July 1992, 440-446.

[2] Burkhard Monien, Ivan H. Sudborough. *Bandwidth Constrained NP-Complete Problems*, Theoretical Computer Science 41 (1985) 141-167.

[3] Christos H. Papadimitriou. *Computational Complexity*, University of California-San Diego, 1994.

[4] David Mitchell, Bart Selman and Hector Levesque. *Hard and Easy Distributions of SAT Problems*, Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92) San Jose, CA, July 1992, 459-465.

[5] Stuart A. Kauffman. *Origins of Order – Self-Organization and Selection in Evolution*, Oxford University Press, New York, 1993.

[6] Martin D. Davis and Elaine J. Weyuker. *Computability, Complexity and Languages*, Academic Press, 1983.

[7] Richard K. Thompson. *Fitness Landscape Investigated*, M.S. Thesis, July 1995.

[8] Thomas H. Cormen, Charles E. Leiserson and Ronald L. Rivest. *Introduction to Algorithms*, The MIT Press, 1990.

[9] Alden Wright and Richard K. Thompson. *The Computational Complexity of NK Fitness Functions*, 1995.

[10] S. A. Kauffman. Adaptation on rugged fitness landscapes. In D. L. Stein, editor, *Lectures in the Sciences of Complexity*, pp. 527-618. Addison Wesley, 1989.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.